Final Report for Period: 06/2009 - 05/2010 **Submitted on:** 07/09/2010
Principal Investigator: Prvulovic, Milos . **Award ID:** 0447783
Organization: GA Tech Res Corp - GIT
Submitted By:
Prvulovic, Milos - Principal Investigator
Title:
CAREER: Architectural Support for Parallel Execution as a Continuum of Transactions (ASPECT)

Project Participants

Senior Personnel

Name: Prvulovic, Milos
Worked for more than 160 Hours: Yes
Contribution to Project:

Post-doc

Graduate Student

Name: Shen, Jianli
Worked for more than 160 Hours: Yes
Contribution to Project:
Development of simulation environemnt, workload characterization

Name: Yan, Chenyu
Worked for more than 160 Hours: Yes
Contribution to Project:
Workload characterization and conversion of parallel workloads to transactional memory.

Name: Naik, Aniket
Worked for more than 160 Hours: Yes
Contribution to Project:
Simulation environment, extension of transactional memory for point-to-point synchronization.

Name: Doudalis, Ioannis
Worked for more than 160 Hours: Yes
Contribution to Project:
Ioannis examined thread scheduling issues in multi-core processors with shared caches.

Name: Raj, Himansu
Worked for more than 160 Hours: Yes
Contribution to Project:
Examined support for virtualization in multicore processors.

Name: Govindaraj, Srihari
Worked for more than 160 Hours: No
Contribution to Project:

Name: Venkataramani, Guru
Worked for more than 160 Hours: Yes
Contribution to Project:

Name: Oh, Jungju

Worked for more than 160 Hours: Yes
Contribution to Project:

Undergraduate Student

Technician, Programmer

Other Participant

Research Experience for Undergraduates

Organizational Partners

Other Collaborators or Contacts

Since January 2007, the PI has been having weekly meetings with researchers at Intel (Chris Hughes and Sanjeev Kumar) who are exploring emerging workloads for many-core processors. The effect of these meetings on this project has been to help determine how much hardware support can realistically be devoted to programmability issues, and also to identify key problems that affect performance of workloads on many-core processors.

In 2008, one of the collaborators from Intel (Sanjeev Kumar) has moved to Facebook, but he has continued to be involved in our weekly coordination meetings until 2009 (when his workload assignments at Facebook significantly diverged from what this project's research).

In 2009, one of the students who worked on this project (Guru Venkataramani) graduated and joined the George Washington University as faculty, but he continued to be involved in our weekly coordination meetings.

Activities and Findings

Research and Education Activities:
First year:

We have extended our simulation infrastructure to simulate a transactional memory system, including a simple timestamp update mechanism. We have also automatically converted the existing Splash-2 benchmarks to use transactional memory, and have rewritten several of these benchmarks to become 'native' transactional memory applications. We have also presented our preliminary work on data race detection using scalar timestamps at HPCA in February of 2006, and have written a paper on point-to-point synchronization and submitted it to ASPLOS 2006 and several subsequent conferences, but it was rejected. Finally, we used the expertise in timestamp management from this project to help improve counter-mode memory encryption which is used to provide security against hardware snoopers (this work will appear in ISCA 2006).

The graduate students involved in this project attained a good understanding of parallel programming principles and their interaction with multiprocessor hardware, and have also gained a very detailed knowledge about transactional memory systems and simulation of multiprocessor systems.

The PI has taught a graduate course on 'Advanced Topics in Computer Architecture', with a primary focus on multiprocessor systems, their programmability, and debugging. Efforts to recruit undergraduate students in this project have resulted in recruiting an undergraduate student to work on improving the simulation speed of parallel systems, but the student has been preparing for this role and his actual work on this project will begin in the second year of the project.

Second year:

Transactional memory has gained a lot of traction in the past year, and others have published several papers that exploit some of the ideas about expressing synchronization described in the original proposal for this project. As a result, we have shifted out focus from providing this basic

functionality to the issues of using the same hardware for TM and other programmability goals. In addition to using he same hardware to exploit both explicit and implicit parallelism (for TM and TLS) as described in the proposal (via explicit order tracking and manipulation), we are also exploring how to reuse other TM support mechanisms. For example, we found that the conflict detection mechanism used for TM and TLS can also be used as a very efficient fine-grain protection mechanism, with potential applications in security and debugging.

The PI taught, for the second time, a graduate course on 'Advanced Issues in Computer Architecture', with the intent of making this a regular course at Georgia Tech (currently it is listed as special problems course). This course is very much aligned with this CAREER project, and introduces students to issues brought into the mainstream by multi-core processors, such as resource sharing and scheduling between processes, debugging support, and new ways to exploit parallelism implicitly of explicitly (such as transactional memory, thread-level speculation, etc.).

In addition to continuing to develop this course, the PI has succeeded in involving an undergraduate student in this CAREER project. The student has been looking into the problem of simulation of multi-threaded execution, with the focus on how to accurately fast forward and sample execution (a la SimPoints) when multiple processes or threads are running and interacting in the same machine. This experience has certainly helped motivate this student to enter the PhD programs this year (he graduated in May 2007). The PI is currently working on recruiting another promising undergraduate student for this project.

Third year:

Many-core processors are increasingly seen as the future mainstream architecture for general-purpose computing. Both researchers and industry are increasingly realizing that programmability issues for many-core processors involve not only correctness, but also scalability of performance. Most of the past work in transactional memory and other mechanisms for programmability of parallel machines has focused on how to make it easier to write correctly synchronized programs. However, little work has been done on how to help programmers write code that is not only correct but also scales well. As a result of this, this project has focused on understanding and exposing to the programmer performance limiters in parallel execution, both explicitly synchronized and transactional.

The PI also continues to include the issues of parallel programmability and performance into the undergraduate and graduate curriculum. In this year of the project, the PI has successfully integrated multi-core architectures and hardware support for programmability into CS 4290/6290, a mainstream senior undergraduate and introductory graduate course that teaches basics of modern computer architecture. In CS 2200, a required sophomore undergraduate course for CS majors, the PI has expanded the emphasis on multi-core processors and added an introduction to multi-core programmability issues.

Fourth year:

Our work on finding scalability bottlenecks in multi-core programs has been expanded to include changing the traditional definitions in computer architecture for use by programmers. In particular, we have identified that the dominant definition of false sharing is very hardware centric and not suitable for programmers who are trying to improve the performance of their programs. False sharing is defined as 'unnecessary' communication among processors, but the definition of 'unnecessary' can loosely be interpreted as 'there exists a possible machine design where this communication would not have occurred'. This definition is useful for designing future machines, but is largely irrelevant to programmers who are concerned with how their program behaves on a given machine or a family of machines. We have redefined the definition of false sharing to account for communication that may be avoidable in principle, but is unavoidable on the particular machine where the program is running. The most obvious difference between prior definitions and ours is in how writes are accounted for. In existing definitions, communication is deemed necessary only when a write on one processor is followed by a read on another processor (because there is real flow of new values from one thread to the other), but an upgrade miss that occurs when a read is followed by a write is considered to be unnecessary (no flow of values, so it may be possible to eliminate the upgrade miss e.g. by using a write-update protocol). The programmer's interpretation of false sharing is usually that it can be avoided by separating the affected variables, e.g. through padding of the data. A consequence of this is that in a real machine that uses a write-invalidate protocol the upgrade miss is necessary ? the thread that suffers the miss is overwriting the same location that was read by another thread, so padding or another technique for alleviation of false sharing cannot eliminate this miss. We have redefined false sharing to account for actual behavior on a given machine, rather than abstract possibilities, and we are working on defining other architectural concepts (load balance, lock contention, etc.) in a similar programmer-friendly way. This work has been submitted to ACM TACO, has been reviewed, and is now undergoing a major revision as requested by the editor.

In the domain of education, the PI has included in his undergraduate class (CS 2200, Introduction to Systems and Networking) assignments that help the students internalize multi- and many-core concepts better. This includes simple parallel programming assignments that require students to correctly implement decomposition of work, synchronization, and load balancing. The key property of these assignments is that they must be complex enough to expose students to targeted concepts, while being simple enough to permit students to complete the assignment within the short allotted time (usually a few or two). In year 4, the PI also taught a graduate class (CS 6290, Advanced Computer Architecture) where a more involved project was introduced to expose students not only to basic concepts needed to produce working parallel code, but also to issues

that affect its parallel scaling (computation-to-communication ratio, load balancing, lock contention, etc). This project appears to have been very beneficial to students (student feedback was very positive) and also provided the PI with an opportunity to learn which mistakes are typically made by inexperienced programmers when writing parallel programs.

Fifth Year:

This year the project activities have focused on extending the performance debugging work beyond mere accounting for events. This work is also part of the PI's new NSF/SRC-funded project (performance debugging support for many-core processors) and is focusing on automatically infering and reporting (in a format programmers can use directly) the causes and 'culprit' lines of code for poor parallel performance.

Another aspect of the project has been to improve rollback and replay support so it can be used for more effective debugging of parallel code, even for bugs whose effects are dormant for long period of time.

The PI's educational activities in year five include teaching CS 4290/6290 (Advanced Computer Architecture) again, with further improvements in project structure and class materials that focus on many-core processors and how they interact with software.

Finally, a significant consideration for a CAREER award is to help the PI in the early development of their academic careers. PI Prvulovic is pleased to report that, over the five-year duration of the project, he has increased his visibility by presenting numerous papers at top conferences, by serving in numerous program committees (about two per year) of top conferences, and by serving in important service roles in the computer architecture community - he is the current Secretary/Treasurer of ACM SIGMICRO, has served as the registration chair for HPCA 2009, and is the finance chair for MICRO 2010.

Within Georgia Tech, the PI has been awarded tenure and promotion to the rank of Associate Professor in 2009. Prior to that, has been elected (by his peers) to represent assistant professors in the newly formed Chair's Advisory Committee - a role he held until he became ineligible for it (promotion our of the assistant professor rank), has served on the PhD student admissions committee, as the moderator for the school-wide annual review of PhD students, etc.

Overall, this CAREER award hs not only helped fund cutting-edge research activity, but has also helped and encouraged the PI to pursue educational and service activities that have been very helpful for his career development.

Findings:

Our major findings thus far are:

1) We found that scalar clocks can be successfully used for both data race detection and order-recording for later deterministic replay. Although we find that most individual data races are not found with scalar clocks, we still find most occurrences of synchronization problems that are causing data races. We also successfully integrated data race detection and order-recording so they use the same set of timestamps to simplify the overall hardware support, and find that simple improvements in how scalar clocks are updated have a dramatic positive impact on synchronization problem detection rates.

2) We found that transactional memory can be easily extended to provide composable and easy-to-use support for point-to-point (conditional) synchronization. Existing transactional memory proposals only target composable and easy-to-use critical sections (atomicity), but do not support conditional (wait for a condition) synchronization. Our new support for conditional synchronization completes transactional memory to allow composable, efficient, and easy-to-use conditional synchronization and conditional critical sections, in addition to transactional memory's existing ability to provide composable, efficient, and easy-to-use critical sections.

3) We found that the mechanisms used to implement transactional memory can also be used for other purposes. For the purposes of our work, we divide TM support mechanisms into three categories: 1) buffering of speculative state, which is used in TM to allow a transaction to be aborted, 2) conflict detection, which is used in TM to detect when two transactions interfere with each other in order to trigger an abort, and 3) conflict resolution, which is triggered upon conflict detection and must abort one or more transactions to ensure consistency and forward progress. If these mechanisms are designed carefully and with enough flexibility, they can also be used for other purposes, which previously needed their own dedicated hardware. In addition to reusing TM mechanisms for Thread Level Speculation (TLS) as described in the original proposal, we found that 1) buffering of speculative state can be used to support efficient checkpointing for reliability and possibly recovery for attacks, and 2) conflict detection with different conflict resolution can be used as a fine-grain protection mechanisms, with applications in security, monitoring, and debugging. Unfortunately, TM support that is thus far present in the literature is designed monolithically, with hard-wired interfaces and interactions between its support mechanisms. This prevents reuse of these mechanisms. We found that each mechanism must have a more clearly defined interface that is controllable by software, to enable a wider variety of applications for that

mechanism.

4) We found that adverse sharing patterns, especially false sharing and complex synchronization, can have major effect on performance, and that this effect is more pronounced with optimistic concurrency control, such as transactional memory, because they have more activity per unit time. We found that relatively simple hardware support can be added to each core to identify key adverse sharing patterns, such as excessive true sharing or false sharing that results in coherence misses, or destructive sharing that results in capacity or conflict misses in shared caches. For each cache miss, our new hardware support allows it to be classified into one of these categories. Existing profiling support, such as performance counters, can then be used to report this data to the programmer in a way that allows the programmer to directly address the problem. Our initial experiences with this support indicate that this classification of problems helps significantly when trying to improve performance of a program, especially on many-core architectures where scalability is a major issue. This is because the type of cache misses behind the performance problem often also points to the method that should be used to address the problem, e.g. false sharing is typically addressed by adding padding between shared data items, true sharing is typically addressed by changing data layout or devising a better division of work among threads, etc.

5) We have preliminary findings indicating that synchronization remains a major scalability limiter for many-core architectures. Some of this problem can be alleviated through transactional memory's optimistic concurrency and low-overhead atomic sections. However, in many large-scale codes the bulk of the problem is caused by waiting on point-to-point synchronization. We are exploring hardware and software mechanisms that would identify the underlying cause of the problem and help programmers improve their code to reduce this waiting time.

6) We found that concepts needed to understand and apply parallelism can be taught effectively at the undergraduate level (primarily juniors and seniors), and that careful crafting of programming exercises can help students internalize these concepts better. We are exploring improvements to this strategy, possibly by including some concepts earlier in the curriculum (freshmen or sophomores).

Training and Development:

Students involved in this project attained a good understanding of parallel programming principles, their interaction with multi- and many-core hardware, and have also gained a very detailed knowledge about simulation of multi-core systems and new programming models such as transactional memory.

The PI has gained a better understanding of our simulation infrastructure, parallel workloads, and the interaction between the programmer, the program, and the hardware. By introducing this material into his undergraduate and graduate teaching, the PI has also gained valuable insight into how programmers interact with tools and mechanisms that can help them improve correctness and performance of their parallel applications.

Of the PhD students who were significantly funded by this award, two have already graduated. One (Chenyu Yan) has been working for Microsoft since graduation. The other (Guru Venkataramani) has joined the faculty at the George Washington University. Guru's decision to pursue an academic career has, at least in part, been a result of mentoring and guidance by the PI.

Outreach Activities:

Journal Publications

Aniket Naik, Chenyu Yan, Milos Prvulovic, "AnySync: Efficient Composable Custom Synchronization in Transactional Memory Systems", 12th International Conference on Architectural Support for Programming Languages and Operating Systems, p. , vol. , (2006). Rejected.,

Guru Venkataramani, Chris Hughes, Sanjeev Kumar, Milos Prvulovic, "DeFT: Design Space Exploration for On-the-Fly Detection of False and True Sharing Misses in Multi-Core Processors", ACM Transactions on Computer Architecture and Compiler Optimization, p. , vol. , (2009). Major Revision,

Venkataramani, G; Doudalis, I; Solihin, Y; Prvulovic, M, "MemTracker: An Accelerator for Memory Debugging and Monitoring", ACM TRANSACTIONS ON ARCHITECTURE AND CODE OPTIMIZATION, p. , vol. 6, (2009). Published, 10.1145/1543753.154375

S. Chhabra, B. Rogers, Y. Solihin, M. Prvulovic, "Making Secure Processors OS- and Performance-Friendly", ACM Transactions on Architecture and Code Optimization (TACO), p. 1, vol. 5, (2009). Published,

Books or Other One-time Publications

I. Doudalis, M. Prvulovic, "HARE: Hardware Assisted Reverse Execution", (2010). Conference Proceedings, Published
Collection: Proceedings of the 16h International Symposium on High-Performance Computer Architecture (HPCA)
Bibliography: ISBN 978-1-4244-5659-8, Published by IEEE

M. Kharbutli, X. Jiang, Y. Solihin, G. Venkataramani, M. Prvulovic, "Comprehensively and Efficiently Protecting the Heap", (2006). Conference Proceedings, Published
Collection: Proceedings of the 12th ACM Int. Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS)
Bibliography: ISBN 1-59593-451-0, Published by ACM Press

B. Rogers, M. Prvulovic, Y. Solihin, "Efficient Data Protection for Distributed Shared Memory Multiprocessors", (2006). Conference Proceedings, Published
Collection: Proceedings of the 15th International Conference on Parallel Architectures and Compilation Techniques (PACT)
Bibliography: ISBN: 1-59593-264-X, Published by ACM Press

Web/Internet Site

URL(s):
http://sourceforge.net/projects/sesc
Description:
This is the SourceForge open-source repository that contains the SESC simulator that has been extended by the PI in the course of this project to support parallel execution under OpenMP and pthreads models.

Other Specific Products

Contributions

Contributions within Discipline:
Multi-core and multi-threaded processors can not be used efficiently without parallel software, and this software is very difficult to write. As a result, support for transactional memory, detection of synchronization problems, and debugging of parallel programs is a critical problem in computer architecture.

Our work on data race detection significantly reduces the hardware complexity needed to support problem detection, and uses the same hardware both for detection of synchronization problems and for order-recording which is needed to achieve effective debugging using deterministic replay.

Our work on conditional synchronization significantly improves the state of the art in programmability of parallel programs. Together with transactional memory's ability to provide composable and easy-to-use critical sections, our work provides a complete solution that allows any synchronization (conditionals, critical sections, or conditional critical sections) to be expressed in a composable and easy-to-use way.

Our work on reusing TM mechanisms for reliability, security, and debugging lowers the effective cost of supporting TM, and provides guidelines for how to design TM mechanisms in a modular fashion so that they can be reused.

Our work on detection of adverse sharing patterns makes it easier to write efficient scalable parallel code. Performance debugging is very hard because a typical programmer does not have a complete understanding of all the mechanisms that affect performance and scalability. As we move towards many-core processors (with tens of cores) scalability will be paramount. By pinpointing the exact patterns that are adversely affecting performance and scalability, we make it easier for the programmer to address this problem.

Contributions to Other Disciplines:

Contributions to Human Resource Development:

The PI has taught a combined undergraduate/graduate class on advanced issues in computer architecture. The class covers important but often neglected aspects of computer architecture, such as reliability, programmability, support for advanced parallel programming models, security, debugging, and virtualization. This class provides a new perspective for students and will be very beneficial to them in the future.

The PI also taught a sophomore-level undergraduate class on systems and networking. This class provides an overview of multi-threaded programming, and an introduction to synchronization. The PI has added new material to the class that also provides students with a more thorough understanding of pitfalls in parallel programming and how to avoid them. Because these students will likely never encounter a single-core processor when they get their jobs, it is imperative to educate them accordingly.

Finally, the PI has taught a class co-taught for senior-level undergraduates and entry-level graduate students. The class is an introduction to modern computer architecture, and the PI has changed the focus of the class towards multi-core architecture. This new focus puts much more emphasis on coherence, sharing of resources among cores, synchronization, and programmability support. Because multi-core and soon many-core architectures are the new mainstream general-purpose computing platform, this course enhancement is essential for keeping these students competitive in the global economy.

Contributions to Resources for Research and Education:

The PI has redesigned the front-end of the SESC simulator to allow simulation of parallel applications based on pthreads and OpenMP. The SESC simulator is increasingly being used by researchers in computer architecture, as evidenced by increasing numbers of SESC citations in top architecture conferences.

The PI has also adapted SESC for use in classes, in particular for simulation of many-core systems to expose students to future many-core architectures before these architectures actually appear on the market. In recent classes the PI has taught, 16- and 32-core simulations were used, which for most of the students in the class was the first time they were exposed to parallelism on that scale.

Contributions Beyond Science and Engineering:

Conference Proceedings

Venkataramani, G;Roemer, B;Solihin, Y;Prvulovic, M, MemTracker: Efficient and programmable support for memory access monitoring and debugging, "FEB 10-14, 2007", THIRTEENTH INTERNATIONAL SYMPOSIUM ON HIGH-PERFORMANCE COMPUTER ARCHITECTURE, PROCEEDINGS, : 273-284 2007

Prvulovic, M, CORD: Cost-effective (and nearly overhead-free) Order-Recording and Data race detection, "FEB 11-15, 2006", TWELFTH INTERNATIONAL SYMPOSIUM ON HIGH-PERFORMANCE COMPUTER ARCHITECTURE, PROCEEDINGS, : 236-247 2006

Yan, CY;Rogers, B;Englender, D;Solihin, Y;Prvulovic, M, Improving cost, performance, and security of memory encryption and authentication, "JUN 17-21, 2006", 33rd International Symposium on Computer Archtiecture, Proceedings, : 179-190 2006

Rogers, B;Chhabra, S;Solihin, Y;Prvulovic, M, Using address independent seed encryption and Bonsai Merkle Trees to make secure processors OS- and performance-friendly, "DEC 01-05, 2007", MICRO-40: PROCEEDINGS OF THE 40TH ANNUAL IEEE/ACM INTERNATIONAL SYMPOSIUM ON MICROARCHITECTURE, : 183-194 2007

Rogers, B;Yan, CY;Chhabra, S;Prvulovic, M;Solihin, Y, Single-Level Integrity and Confidentiality Protection for Distributed Shared Memory Multiprocessors, "FEB 16-20, 2008", 2008 IEEE 14TH INTERNATIONAL SYMPOSIUM ON HIGH PEFORMANCE COMPUTER ARCHITECTURE, : 149-160 2008

Venkataramani, G;Doudalis, I;Solihin, Y;Prvulovic, M, FlexiTaint: A Programmable Accelerator for Dynamic Taint Propagation, "FEB 16-20, 2008", 2008 IEEE 14TH INTERNATIONAL SYMPOSIUM ON HIGH PEFORMANCE COMPUTER ARCHITECTURE, : 161-172 2008

Categories for which nothing is reported:

Organizational Partners

Activities and Findings: Any Outreach Activities

Any Product

Contributions: To Any Other Disciplines

Contributions: To Any Beyond Science and Engineering

Georgia Tech College of Computing
School of Computer Science

February 26, 2009

To Whom It May Concern:

This letter serves to re-affirm the School of Computer Science in the College of Computing at the Georgia Institute of Technology continues its commitment to mentor Prof. Milos Prvulovic, and to support the integrated research and educational activities and the work plan for the project "CAREER: Architectural Support for Parallel Execution as a Continuum of Transactions (ASPECT)".

As the Chair of this School, I have read and I continue to endorse this career development plan. I attest that the PI's career-development plan is supported by and integrated into the educational and research goals of this School, the College of Computing, and Georgia Tech. I personally commit to the support and professional development of the PI.

Please do not hesitate to contact me directly should you have any further questions.

Sincerely,

Ellen W. Zegura, D.Sc.
Chair and Professor
School of Computer Science
Georgia Tech

266 Ferst Drive • Atlanta, GA 30332-0765 USA • Phone 404.894.6711 • Fax 404.385.6506 • www.cc.gatech.edu